sSCADA: securing SCADA infrastructure communications

Yongge Wang

Department of SIS,
UNC Charlotte,
9201 University City Blvd,
Charlotte, NC 28223, USA
E-mail: yonwang@uncc.edu

Abstract: Distributed control systems (DCS) and supervisory control and data acquisition (SCADA) systems were developed to reduce labour costs, and to allow system-wide monitoring and remote control from a central location. Control systems are widely used in critical infrastructures such as electric grid, natural gas, water and wastewater industries. While control systems can be vulnerable to a variety of types of cyber attacks that could have devastating consequences, little research has been done to secure the control systems. American Gas Association (AGA), IEC TC57 WG15, IEEE, NIST and National SCADA Test Bed Program have been actively designing cryptographic standard to protect SCADA systems. American Gas Association (AGA) had originally been designing cryptographic standard to protect SCADA communication links and finished the report AGA 12 part 1. The AGA 12 part 2 has been transferred to IEEE P1711. This paper presents an attack on the protocols in the first draft of AGA standard (Wright et al., 2004). This attack shows that the security mechanisms in the first version of the AGA standard protocol could be easily defeated. We then propose a suite of security protocols optimised for SCADA/DCS systems which include: point-to-point secure channels, authenticated broadcast channels, authenticated emergency channels, and revised authenticated emergency channels. These protocols are designed to address the specific challenges that SCADA systems have.

Keywords: supervisory control and data acquisition; SCADA; distributed control systems; DCS; cyber attacks; smart grid security; critical infrastructure protection; secure communication

Reference to this paper should be made as follows: Wang, Y. (2011) ‘sSCADA: securing SCADA infrastructure communications’, *Int. J. Communication Networks and Distributed Systems*, Vol. 6, No. 1, pp.59–78.

Biographical notes: Yongge Wang received his PhD from the University of Heidelberg, Germany. Since then, he had worked in the industry for a few years until he joined UNC Charlotte in 2002. He has actively participated and contributed to the standards bodies such as IETF, W3C XML Security protocols, IEEE 1363 standardisation groups for cryptographic techniques and ANSI T11 groups for SAN network security standards. He is the inventor of Remote Password Authentication Protocol SRP5 (an IEEE 1363.2 Standard) and identity-based key agreement protocol WANG-KE (an IEEE 1363.3 Standard).

1 Introduction

Control systems are computer-based systems that are used within many critical infrastructures and industries (e.g., electric grid, natural gas, water and wastewater industries) to monitor and control sensitive processes and physical functions. Without a secure supervisory control and data acquisition (sSCADA) system it is impossible to protect the nation's critical infrastructures.

Typically, control systems collect sensor measurements and operational data from the field, process and display this information, and relay control commands to local or remote equipments. Control systems may perform additional control functions such as operating railway switches, circuit breakers, and adjusting valves to regulate flow in pipelines. The most sophisticated ones control devices and systems at an even higher level.

Control systems have been in place since the 1930s and there are two primary types of control systems. Distributed control systems (DCS) and supervisory control and data acquisition (SCADA) systems. DCS systems typically are used within a single processing or generating plant or over a small geographic area. SCADA systems typically are used for large, geographically dispersed distribution operations. For example, a utility company may use a DCS to generate power and a SCADA system to distribute it. We will concentrate on SCADA systems and our discussions are generally applicable to DCS systems.

In a typical SCADA system (Cegrell, 1986), data acquisition and control are performed by remote terminal units (RTU) and field devices that include functions for communications and signalling. SCADA systems normally use a poll response model for communications with clear text messages. Poll messages are typically small (less than 16 bytes) and responses might range from a short 'I am here' to a dump of an entire day's data. Some SCADA systems may also allow for unsolicited reporting from remote units. The communications between the control centre and remote sites could be classified into following four categories.

1 *Data acquisition*: the control centre sends poll (request) messages to RTU and the RTU dump data to the control centre. In particular, this includes *status scan and measured value scan*. The control centre regularly sends a status scan request to remote sites to get field devices status (e.g., open or closed or a fast closed-open-closed sequence) and a measured value scan request to get measured values of field devices. The measured values could be analogue values or digitally coded values and are scaled into engineering format by the front-end processor (FEP) at the control centre.

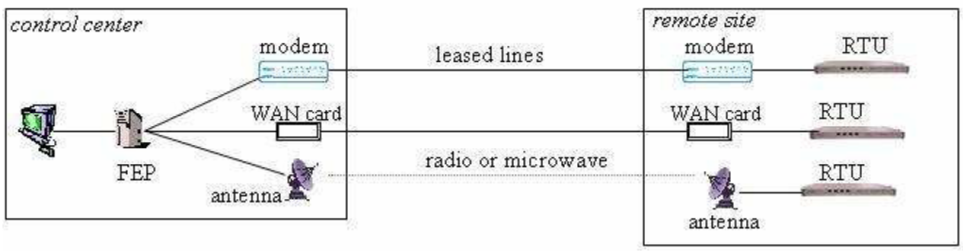
2 *Firmware download*: the control centre sends firmware downloads to remote sites. In this case, the poll message is larger (e.g., larger than 64K bytes) than other cases.

3 *Control functions*: the control centre sends control commands to a RTU at remote sites. Control functions are grouped into four subclasses: individual device control (e.g., to turn on/off a remote device), control messages to regulating equipment (e.g., a raise/lower command to adjust the remote valves), sequential control schemes (a series of correlated individual control commands), and automatic control schemes (e.g., closed control loops).

4 *Broadcast*: the control centre may broadcast messages to multiple RTU. For example, the control centre broadcasts an emergent shutdown message or a set-the-clock-time message.

Acquired data is automatically monitored at the control centre to ensure that measured and calculated values lie within permissible limits. The measured values are monitored with regard to rate-of-change and for continuous trend monitoring. They are also recorded for post-fault analysis. Status indications are monitored at the control centre with regard to changes and time tagged by the RTU. In legacy SCADA systems, existing communication links between the control centre and remote sites operate at very low speeds (could be on an order of 300 bps to 9,600 bps). Note that present deployments of SCADA systems have variant models and technologies, which may have much better performances (for example, 61,850-based systems). Figure 1 describes a simple SCADA system.

Figure 1 A simple SCADA system (see online version for colours)



In practice, more complicated SCADA system configurations exist. Figure 2 lists three typical SCADA system configurations (see, e.g., AGA Report No. 12, 2004).

Recently, there have been several efforts to secure the national SCADA systems. The examples are:

1 American Gas Association (AGA) (AGA Report No. 12, 2004). AGA is among the first to design cryptographic standard to protect SCADA systems. AGA had originally been designing cryptographic standard to protect SCADA communication links and finished the report AGA 12 part 1. The AGA 12 part 2 has been transferred to IEEE P1711.

2 IEEE P1711. This is transferred from AGA 12 part 2. This standard effort tries to define a security protocol, the Serial SCADA Protection Protocol (SSPP), for control system serial communication.

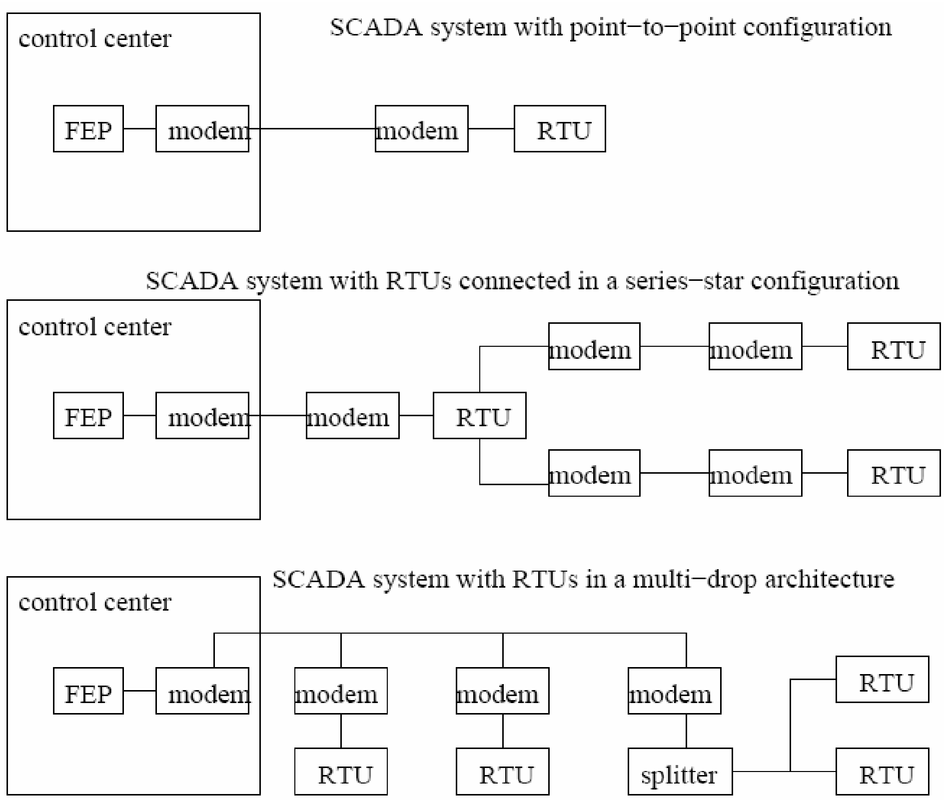
3 IEEE P1815. Standard for Electric Power Systems Communications – Distributed Network Protocol (DNP3). The purpose of this standard is to document and make available the specifications for the DNP3 protocol.

4 IEC TC57 WG15. IEC TC57 WG57 standardise SCADA communication security via its IEC 608705 series.

5 NIST (2008). The NIST Industrial Control System Security (ICS) group works on general security issues related to control systems such as SACAD systems.

6 National SCADA Test Bed Program (Idaho National Laboratory, 2008). The Department of Energy established the National SCADA Test Bed Program at Idaho National Laboratory and Sandia National Laboratory to ensure the secure, reliable and efficient distribution of power.

Figure 2 Typical SCADA system configurations



2 Threats to SCADA systems

Several (real and simulated) attacks on SCADA systems were reported in the past few years (Abrams and Weiss, 2007; *USA Today*, 2007). In the Maroochy Shire attack (Abrams and Weiss, 2007), an Australian man hacked into the Maroochy Shire, Queensland computerised waste management system and caused millions of litres of raw sewage to spill out into local parks, rivers and even the grounds of a Hyatt Regency hotel. It is reported that the 49-year-old Vitek Boden had conducted a series of electronic attacks on the Maroochy Shire sewage control system after his job application had been rejected. Later investigations found radio transmitters and computer equipments in Boden's car. The laptop hard drive contained software for accessing and controlling the sewage SCADA systems. The simulated Aurora attack (*USA Today*, 2007) conducted in March 2007 by the US Department of Homeland Security resulted in the partial destruction of a $1 million dollar large diesel-electric generator.

SCADA systems were not designed with public access in mind; they typically lack even rudimentary security. However, with the advent of technology and particularly the internet, much of the technical information required to penetrate these systems is widely discussed in the public forums of the affected industries. Critical security flaws for SCADA systems are well known to potential attackers. It is feared that SCADA systems can be taken over by hackers, criminals, or terrorists. Some companies may assume that they use leased lines and therefore nobody has access to their communications. The fact is that it is easy to tap these lines [available at http://www.tscm.com/outsideplant.html (accessed on 22 February 2010)]. Similarly, frequency hopping spread spectrum radio and other wireless communication mechanisms frequently used to control RTU can be compromised as well.

Several efforts (GAO-04-628T, 2004; NIST, 2008; Idaho National Laboratory, 2008) have been put on the analysis and protection of SCADA system security. According to these reports (GAO-04-628T, 2004; NIST, 2008; Idaho National Laboratory, 2008), the factors that have contributed to the escalation of risk to SCADA systems include:

- The adoption of standardised technologies with known vulnerabilities. In the past, proprietary hardware, software and network protocols made it difficult to understand how SCADA systems operated – and therefore how to hack into them. Today, standardised technologies such as Windows, Unix-like operating systems, and common internet protocols are used by SCADA systems. Thus the number of people with knowledge to wage attacks on SCADA systems have increased.
- The connectivity of control systems to other networks. In order to provide decision makers with access to real-time information and allowing engineers to monitor and control the SCADA systems from different points on the enterprise networks, the SCADA systems are normally integrated into the enterprise networks. Enterprises are often connected to partners' networks and to the internet. Some enterprises may also use wide area networks and internet to transmit data to remote locations. This creates further security vulnerabilities in SCADA systems.
- Insecure remote connections. Enterprises often use leased lines, wide area networks/internet, and radio/microwave to transmit data between control centres and remote locations. These communication links could be easily hacked.
- The widespread availability of technical information about control systems. Public information about infrastructures and control systems is readily available to potential hackers and intruders. For example, Sean Gorman's dissertation (see, e.g., Blumenfeld, 2003; Rappaport, 2007) mapped every business and industrial sector in the US economy to the fibre-optic network that connects them, using materials that was available publicly on the internet. In addition, significant information on SCADA systems is publicly available (from maintenance documents, from former employees, and from support contractors, etc.). All these information could assist hackers in understanding the systems and to find ways to attack them.

Hackers may attack SCADA systems with one or more of the following actions.

1 denial of service attacks by delaying or blocking the flow of information through control networks

2 make unauthorised changes to programmed instructions in RTU at remote sites, resulting in damage to equipment, premature shutdown of processes, or even disabling control equipment

3 send false information to control system operators to disguise unauthorised changes or to initiate inappropriate actions by system operators

4 modify the control system software, producing unpredictable results

5 interfere with the operation of safety systems.

The analysis in reports such as GAO-04-628T (2004), NIST (2008) and Idaho National Laboratory (2008) show that securing control systems poses significant challenges which include

1 The limitations of current security technologies in securing control systems. Existing internet security technologies such as authorisation, authentication and encryption require more bandwidth, processing power and memory than control system components typically have. Controller stations are generally designed to do specific tasks, and they often use low-cost, resource-constrained microprocessors.

2 The perception that securing control systems may not be economically justifiable.

3 The conflicting priorities within organisations regarding the security of control systems.

In this paper, we will concentrate on the protection of SCADA remote communication links. In particular, we discuss the challenges on protection of these links and design new security technologies to sSCADA systems.

3 Securing SCADA remote connections

Relatively cheap attacks could be mounted on SCADA system communication links between the control centre and RTU since there is neither authentication nor encryption on these links. Under the umbrella of NIST 'Critical infrastructure protection cybersecurity of industrial control systems', 'AGA SCADA Encryption Committee' has been trying to identify the functions and requirements for authenticating and encrypting SCADA communication links. Their proposal AGA Report No. 12 (2004) is to build cryptographic modules that could be invisibly embedded into existing SCADA systems (in particular, one could attach these cryptographic modules to modems of Figure 2) so that all messages between modems are encrypted and authenticated when necessary, and they have identified the basic requirements for these cryptographic modules. However, due to the constraints of SCADA systems, no viable cryptographic protocols have been identified to meet these requirements. In particular, the challenges for building these devices are (see AGA Report No. 12, 2004):

1 encryption of repetitive messages

2 minimising delays due to cryptographic operations

3 assuring integrity with minimal latency

- intra-message integrity: if cryptographic modules buffer message until the message authenticator is verified, it introduces message delays that are not acceptable in most cases
- inter-message integrity: reorder messages, replay messages and destroy specific messages

4 accommodating various SCADA poll-response and retry strategies: delays introduced by cryptographic modules may interfere with the SCADA system's error-handling mechanisms (e.g., time-out errors)

5 supporting broadcast messages

6 incorporating key management

7 cost of device and management

8 mixed mode: some SCADA systems have cryptographic capabilities while others not

9 accommodate to different SCADA protocols: SCADA devices are manufactured by different vendors with different proprietary protocols.

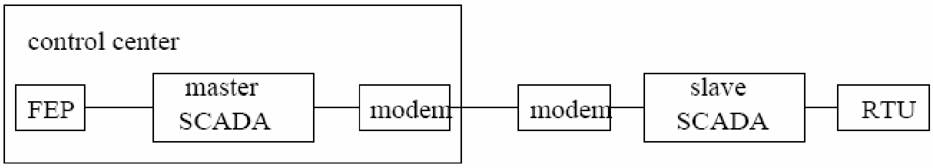
This paper designs efficient cryptographic mechanisms to address these challenges and to build cryptographic modules as recommended in AGA Report No. 12 (2004). These mechanisms can be used to build plug-in devices called sSCADA that could be inserted into SCADA networks so that all communication links are authenticated and encrypted. In particular, authenticated broadcast protocols are designed so that they can be cheaply included into these devices. It has been a major challenging task to design efficiently authenticated emergency broadcast protocols in SCADA systems.

The trust requirements in our security protocol design are as follows. RTU devices are deployed in untrusted environments and individual remote devices could be controlled by adversaries. The communication links are not secure but messages (maybe modified or re-ordered) could be delivered to the destination with certain probability. In another word, complete denial of service attacks (e.g., jamming) on the communication links are not addressed in our protocol. Compromising the control centre in a SCADA system will make the entire system useless. Thus we assume that control centres are trusted in our protocol.

4 sSCADA protocol suite

The sSCADA protocol suite is proposed to overcome the challenges that we have discussed in the previous section. sSCADA devices that are installed at the control centre is called master sSCADA device, and sSCADA devices that are installed at remote sites are called slave sSCADA devices. Each master sSCADA device may communicate privately with several slave sSCADA devices. Once in a while, the master sSCADA device may also broadcast authenticated messages to several slave sSCADA devices (e.g., an emergency shutdown). An illustrative sSCADA device deployment for point-to-point SCADA configuration is shown in Figure 3.

Figure 3 sSCADA with point-to-point SCADA configuration



4.1 Vulnerabilities of a proposed protocol to AGA

In this section, we discuss vulnerabilities of a proposed protocol to AGA. This analysis shows the challenges in designing secure communication protocols for SCADA systems. A point to point secure channel protocol has been proposed by the AGA standard draft (AGA Report No. 12, 2004; Wright et al., 2004) (an open source implementation could be found at *SCADAsafe*). We first briefly review this protocol in the following.

Preshared secrets are installed into the master sSCADA and slave sSCADA devices during deployment. These secrets are used to negotiate session encryption and authentication keys for the two devices. Each sSCADA device maintains a send sequence state variable in order to assign a sequence number to each ciphertext message it sends. The send sequence variable is initialised to one at session negotiation and incremented with every ciphertext message sent. Let i be the current send sequence number and $P = p_1 \ldots p_n$ be the plaintext message that the sSCADA device wants to send, where $p_j (j = 1, \ldots, n)$ are blocks of the cipher block length (for example, if AES128 is used, then p_j contains 128 bits). Then the sending sSCADA device enciphers P to the ciphertext C as follows:

$$C = ic_1, c_2 \cdots c_n a$$

where

$$c_j = E_k[p_j \oplus E_k[i, j, 00 \ldots]]$$

$$a = MAC_{k'}[iP]$$

$E_k[\cdot]$ denotes the encryption process using the key k, and $MAC_{k'}[\cdot]$ denotes the message authenticator computation process using the key k'. The sending sSCADA device then sends C to the receiving sSCADA device. Let $\overline{C} = \overline{i}\overline{c}_1\overline{c}_2 \cdots \overline{c}_n$ be the message that the receiving sSCADA device receives.

At the receiving side, the sSCADA device maintains a receive sequence state variable in order to record the sequence number of the last authenticated message that it received. The receive sequence variable is initialised to zero at session negotiation. Before decrypting the received ciphertext, the sSCADA device checks that the sequence number $\overline{i}$ contained in the message is greater than the sSCADA's receive sequence variable. If it is not, the sSCADA device discards the remainder of the message. This check is used to ensure that an adversary cannot replay old messages (in the following, our analysis shows that this protection could be easily defeated). Provided the sequence number check succeeds, the receiving sSCADA device decrypts the message as follows:

$$\overline{P} = \overline{p}_1\overline{p}_2 \cdots \overline{p}_n$$

where

$\overline{p}_j = D_k[\overline{c}_j] \oplus E_k[\overline{i}, j, 00\ldots].$

The receiving sSCADA device forwards the decrypted plaintext block $\overline{p}_j$ to the SCADA system as soon as they are available. Finally, the receiving sSCADA device computes the message authentication code (MAC) for the message as follows:

$$\tilde{a} = \mathrm{MAC}_{k'}[\overline{i}P]$$

and compares it to the MAC $\overline{a}$. If the two match, the sSCADA device updates its receive sequence variable to the sequence number $\overline{i}$ of the received message, and otherwise it logs an error.

Now we present our attack on the above protocol in the following. Assume that the adversary Carol controls the communication links between the sending and receiving sSCADA devices and the current receive sequence state variable at the receiving sSCADA side contains the value i_0. When the sending sSCADA device sends the message ciphertexts $C = ic_1c_2 \cdot \cdot \cdot c_na$ for $i > i_0$ in the future, Carol forwards these ciphertexts to the receiving sSCADA device by modifying one bit in the authenticators a (all other bits are forwarded as it is). When the receiving sSCADA devices receive these ciphertexts, it checks the sequence numbers (which are correct), decrypts the ciphertext blocks, and forwards the decrypted plaintext blocks to the SCADA system. However, since the authenticators have been tampered, the receiving sSCADA device fails to check the authenticators. Thus the receiving sSCADA device will only log errors without updating its receive sequence state variable. That is, the receiving sSCADA device will hold the value i_0 for its receive sequence state variable. At the same time, Carol logs all these ciphertexts and observes what happens in the SCADA system. Thus she can learn the meanings of these ciphertexts to the SCADA system. At some time in the future, Carol wants the SCADA system to behave according to the ciphertext C which contains a sequence number larger than i_0. Carol can then just forward this ciphertext to the receiving sSCADA device. Of course, Carol can also tamper the authenticator so that the receiving sSCADA device still holds the value i_0 in its receive state variable after processing this message. The receiving sSCADA device will just decrypt this ciphertext and forward it to the SCADA system since the sequence number contained in C is larger than i_0. In another word, the SCADA system is now in the complete control of Carol's hand.

Another potential pitfall in the proposed protocol is that same keys are used by two sides. This leaves the door open for the attacker to replay the message from one direction in the other direction. This vulnerability could easily be fixed by using different padding schemes or using different keys for different directions. The original authors of the protocol have recommended some fix in *SCADAsafe* to avoid our above attacks.

4.2 Point-to-point secure channels

In the previous section, we presented an attack on the first draft of the AGA proposal. Though the protocol in the AGA proposal could be fixed, in the following, we present a new secure solution. In order to reduce the cost of sSCADA devices and management, only symmetric key cryptographic techniques is used in our design. Indeed, due to the slow operations of public key cryptography, public key cryptographic protocols could introduce delays in message transmission which are not acceptable to SCADA protocols. Semantic security property (Goldwasser and Michali, 1984) is used to ensure that an eavesdropper has no information about the plaintext, even if it sees multiple encryptions of the same plaintext. For example, even if the attacker has observed the ciphertexts of 'shut down' and 'turn on', it will not help the attacker to distinguish whether a new ciphertext is the encryption of 'shut down' or 'turn on'. In practice, the randomisation technique is used to achieve this goal. For example, the message sender may prepend a random string (e.g., 128 bits for AES-128) to the message and use special encryption modes such as chaining block cipher mode (CBC) or Hash-CBC mode (HCBC). In some mode, this random string is called the initialisation vector (IV). This prevents information leakage from the ciphertext even if the attacker knows several plaintext/ciphertext pairs encrypted with the same key.

Since SCADA communication links could be as low as 300 bps and immediate response are generally required, there is no sufficient bandwidth to send the random string (IV) each time with the ciphertext, thus we need to design different cryptographic mechanisms to achieve semantic security without additional transmission overhead. In our design, we use two counters shared between two communicating partners, one for each direction of communication.

The counters are initially set to zeros and should be at least 128 bits, which ensures that the counter values will never repeat; avoiding replay attacks. The counter is used as the IV in message encryptions if CBC or HCBC mode is used. After each message encryption, the counter is increased by one if CBC mode is used and it is increased by the number of blocks of encrypted data if HCBC mode is used. The two communicating partners are assumed to know the values of the counters and the counters do not need to be added to each ciphertext. Messages may get lost and the two counters need to be synchronised once a while (e.g., at off-peak time). A simple counter synchronisation protocol is proposed for the sSCADA protocol suite. The counter synchronisation protocol could also be initiated when some encryption/decryption errors appear due to unsynchronised counters.

In order for two sSCADA devices to establish a secure channel, a master secret key needs to be bootstrapped into the two devices at the deployment time (or when a new sSCADA device is deployed into the existing network). For most configurations, secure channels are needed only between a master sSCADA device and a slave sSCADA device. For some configurations, secure channels among slave sSCADA devices may be needed also. The secure channel identified with this master secret is used to establish other channels such as session secure channels, time synchronisation channels, authenticated broadcast channels, and authenticated emergency channels.

Assume that $\mathcal{H}(\cdot)$ is a pseudorandom function (e.g., constructed from SHA-256) and two sSCADA devices A and B share a secret $\mathcal{K}_{AB} = \mathcal{K}_{BA}$. Depending on the security policy, this key $\mathcal{K}_{AB}$ could be the shared master secret or a shared secret for one session which could be established from the shared master key using a simple key establishment protocol (in order to achieve session key freshness, typically one node sends a random nonce to the other one and the other node sends the encrypted session key together with an authenticator on the ciphertext and the random nonce). Keys for different purposes could be derived from this secret as follows (it is not a good practice to use the same key for different purposes). For example, $\mathcal{K}_{AB} = \mathcal{H}(\mathcal{K}_{AB}, 1)$ is for message encryption from A to B, $K'_{AB} = \mathcal{H}(\mathcal{K}_{AB}, 2)$ is for message authentication from A to B, $K_{BA} = \mathcal{H}(\mathcal{K}_{AB}, 3)$ is for message encryption from B to A, and $K'_{BA} = \mathcal{H}(\mathcal{K}_{AB}, 4)$ is for message authentication from B to A.

Optional MAC is used for two parties to achieve data authentication and integrity. MAC that could be used for sSCADA implementation includes HMAC (Bellare et al., 1996; Krawczyk et al., 1997), CBC-MAC (NIST, 1981), and others. When party A wants to send a message m to party B securely, A computes the ciphertext $c = \mathcal{E}(C_A, K_{AB}, \overline{c}_A \| m)$ and message authenticator $mac = MAC(K'_{AB}, C_A \| c)$, where $\overline{c}_A$ is the last l bits of $\mathcal{H}(C_A)$ (l could be as large as possible if bandwidth is allowed and 32 bits should be the minimal), $\mathcal{E}(C_A, K_{AB}, \overline{c}_A \| m)$ denotes the encryption of $\overline{c}_A \| m$ using key K_{AB} and random-prefix (or IV) C_A and C_A is the counter value for the communication from A to B. Then A sends the following packets to B:

$$A \to B : c, mac \text{ (optional)}$$

When B receives the above packets, B decrypts c, checks that $\overline{c}_A$ is correct, and verifies the message authenticator *mac* if *mac* is present. As soon as B receives the first block of the ciphertext, B can check whether $\overline{c}_A$ is correct. If it is correct, then B continues the decryption and updates its counter. Otherwise, B discards the entire ciphertext. If the message authenticator code *mac* is present, B also verifies the correctness of *mac*. If *mac* is correct, B does nothing, otherwise, B may choose to inform A that the message was corrupted or try to re-synchronise the counters.

There are several implementation issues on how to deliver the message to the target (e.g., RTU). For example, we give a few cases in the following.

1 B uses the counter to decrypt the first block of the ciphertext, if the first l bits of the decrypted plaintext is not consistent with $\mathcal{H}(C_A)$, then the reason could be that the counter C_A is not synchronised or that the ciphertext is corrupted. B may try several possible counters until the counter checking process succeeds. B then uses the verified counter and the corresponding key to decrypt the message and deliver each block of the resulting message to the target as soon as it is available. If no counter could be verified in a limited number of trials, B may notify A of the transmission failure and initiate the counter synchronisation protocol in the next section. The advantage of this implementation is that we have minimised delay from the cryptographic devices, thus minimise the interference of SCADA protocols. Note that in this implementation, the message authenticator *mac* is not used at all. If the ciphertext was tampered, we rely on the error correction mechanisms (normally CRC codes) in SCADA systems to discard the entire message. If CBC (respectively HCBC) mode is used, then the provable security properties (respectively, provable online cipher security properties) of CBC mode (respectively HCBC mode) (Bellare et al., 2000, 2001) guarantees that the attacker has no chance to tamper the ciphertext so that the decrypted plaintext contains correct CRC that was used by SCADA protocols to achieve integrity.

2 Proceed as in the above case 1. In addition, the *mac* is further checked and the decrypted message is delivered to the SCADA system only if the *mac* verification passes. The disadvantage for this implementation is that these cryptographic operations introduce significant delay for message delivery and it may interfere with SCADA protocols.

3 Proceed as in the above case 1. The decrypted message is delivered to the SCADA system as soon as they are available. After receiving the entire message and *mac*, *B* will also verify *mac*. If the verification passes, *B* will do nothing. Otherwise, *B* re-synchronises the counter with *A* or initiates some other exception handling protocols.

4 In order to avoid delays introduced by cryptographic operations and to check the *mac* at the same time, sSCADA devices may deliver decrypted bytes immediately to the target except the last byte. If the message authenticator *mac* is verified successfully, the sSCADA device delivers the last byte to the target. Otherwise, the sSCADA device discards the last byte or sends a random byte to the target. That is, we rely on the error correction mechanisms at the target to discard the entire message. Similar mechanisms have been proposed in AGA Report No. 12 (2004). However, an attacker may insert garbage between the ciphertext and *mac* thus it trick the sSCADA device to deliver the decrypted messages to the SCADA system. If this happens, we essentially do not get advantage from this implementation. Thus this implementation is not recommended.

5 Instead of prepend $\overline{c}_A$ to the plaintext message, one may choose to prepend three bytes of other specially formatted string to the plaintext message (three bytes bandwidth is normally available in SCADA systems) before encryption. This is an acceptable solution though we still prefer our solution of prepending the hash outputs of the counter.

There could be other implementations to improve the performance and interoperability with SCADA protocols. sSCADA device should provide several possible implementations for users to configure. Indeed, sSCADA devices may also be configured in a dynamic way that for different messages it uses different implementations.

In some SCADA communications, message authentication-only is sufficient. That is, it is sufficient for *A* to send (*m*, *mac*) to *B*, where *m* is the cleartext message and $mac = MAC(K'_{AB}, C_A \| m)$. sSCADA device should provide configuration options to do message authentication without encryption. In this case, even if the counter value is not used as the IV, the counter value should still be authenticated in the *mac* and be increased after the operation. This will provide message freshness assurance and avoid replay attacks. sSCADA should also support message pass-through mode. That is, message is delivered without encryption and authentication. In a summary, it should be possible to configure an sSCADA device in such a way that some messages are authenticated and encrypted, some messages are authenticated only, and some messages are passed through directly.

It is straightforward to show that our point-to-point secure channels provide data authentication, data integrity, data confidentiality, and weak data freshness (i.e., messages arrive at the destination in the same order that was sent from the source).

4.3 Counter synchronisation

In the point-to-point message authentication and encryption protocol, we assume that both sSCADA devices *A* and *B* know each other's counter values C_A and C_B. In most cases, reliable communication in SCADA systems is provided and the security protocols in the previous section work fine. Still we provide a counter synchronisation protocol so that sSCADA devices could synchronise their counters when necessary. The counter synchronisation protocol could be initiated by either side. Assume that A initiates the counter synchronisation protocol. Then the protocol looks as follows:

$$A \rightarrow B : N_A$$

$$B \rightarrow A : C_B, MAC(K'_{BA}, N_A \| C_B)$$

This counter synchronisation protocol is analogous to that in Perrig et al. (2002).

The initial counter values of two sSCADA devices could be bootstrapped directly. The above counter synchronisation protocol could also be used by two devices to bootstrap the initial counter values. A master sSCADA device may also use the authenticated broadcast channel that we will discuss in the next section to set several slave sSCADA devices' counters to the same value using one message.

4.4 Authenticated broadcast channels

Encryption and authentication alone are not sufficient for SCADA applications. For example, it is not acceptable to authenticate a message individually in an emergent shutdown when timely responses from the RTU's are critical. In order to support authenticated broadcast, we use one way key chains. This channel can be used to establish other channels such as authenticated emergency channels (see next section).

Typical authenticated broadcast channels require asymmetric cryptographic techniques; otherwise any compromised receiver could forge messages from the sender. Cheung (1997) proposed a symmetric cryptography based source authentication technique in the context of authenticating communication among routers. Cheung's technique is based on delayed disclosure of keys by the sender. Later, it was used in the Guy Fawkes protocol (Anderson et al., 1998) for interactive unicast communication, and in Bergadano et al. (2000a, 2000b), Briscoe (2000) and Perrig et al. (2000, 2001) for streamed data multicast. Perrig et al. (2000, 2001) adapted delayed key disclosure based TESLA protocols to sensor networks for sensor broadcast authentication (the new adapted protocol is called μTESLA). One-way key chains used in these protocols are analogous to the one-way key chains introduced by Lamport (1981) and the S/KEY authentication scheme (Haller, 1995).

In the following, we briefly describe the authenticated broadcast scheme for SCADA systems. At the sender (normally the master sSCADA device or a computer connected to it) set up time, the sender generates a one-way key chain in the setup phase. In order to generate a one-way key chain of length n, the sender chooses a random key $\mathcal{K}_n$ first, then it applies the pseudorandom function $\mathcal{H}$ repeatedly to $\mathcal{K}_n$ to generate the remaining keys. In particular, for each $i < n$, $\mathcal{K}_i = \mathcal{H}(\mathcal{K}_i + 1)$.

For the purpose of broadcast authentication, the sender splits the time into even intervals I_i. The duration of each time interval is denoted as δ (e.g., $\delta = 5$ seconds or 5 minutes or even 2 hours), and the starting time of the interval I_i is denoted as t_i. In another word, $t_i = t_0 + i\delta$. At time t_i, the sender broadcasts the key $\mathcal{K}_i$. Any device that has an authentic copy of key $\mathcal{K}_{i-1}$ can verify the authenticity of the key $\mathcal{K}_i$ by checking whether $\mathcal{K}_{i-1} = \mathcal{H}(\mathcal{K}_i)$. Indeed, any device that has an authentic copy of some key $\mathcal{K}_\upsilon$ ($\upsilon < i$) can verify the authenticity of key $\mathcal{K}_i$ since $\mathcal{K}_\upsilon = \mathcal{H}^{(i-\upsilon)}(\mathcal{K}_i)$.

Let d (a unit of time intervals) be the key disclosure delay factor. The value of d is application dependent and could be configured at deployment time or after deployment (e.g., using the secure broadcast protocols itself). After d is fixed, the sender will use keying materials derived from key $\mathcal{K}_{i+d}$ to authenticate broadcast messages during the time interval I_i. Thus the message being broadcast during time interval I_i could be verified by the receiver during the time interval I_{i+d} after the sender broadcasts $\mathcal{K}_{i+d}$ at time t_{i+d}. It is easy to see that in order to achieve authenticity, the sender and the receiver need to be loosely time synchronised. Otherwise, if the receiver time is slower than the sender's time, an attacker can use published keys to impersonate the sender to the receiver. Typically the key disclosure delay should be greater than any reasonable round trip time between the sender and the receiver. If the sender does not broadcast data frequently, the key disclosure delay may be significantly larger. For example, $d\delta$ could take the value of several hours for some SCADA systems.

If a receiver (typically a slave sSCADA device) is deployed at some time during the interval I_i, the sender needs to bootstrap key $\mathcal{K}_i$ on the one-way key chain to the receiver. The sender also needs to bootstrap the key disclosure schedule which includes the starting time t_i of the time interval I_i, the key disclosure delay factor d, and the duration δ of each time interval. All these information could be bootstrapped to the receiver using the point-to-point secure channel that we have designed in the previous section or using other channels such as manual input. During a time interval I_j ($j > i$), the receiver receives the broadcast key $\mathcal{K}_j$ from the sender and verifies whether $\mathcal{K}_{j-1} = \mathcal{H}(\mathcal{K}_j)$. If the verification is successful, the receiver updates its key on the one-way key chain. If the receiver does not receive the broadcast key during the time interval I_j (either due to packet loss or due to active denial of service attacks such as jamming attacks), it can update its key in the next time interval I_{j+1}.

When a receiver gets a packet from the sender, it first checks whether the key used for the packet authentication has been revealed. If the answer is yes, then the attacker knows the key also and the packet could be a forged one. Thus the receiver needs to discard the packet. If the key have not been revealed yet, the receiver puts the packet in the buffer and checks the authenticity of the packet when the corresponding key is revealed. As stated above, if the sender and the receiver agree on the key disclosure schedule and the time is loosely synchronised, then message authenticity is guaranteed. However, the protocol does not provide non-repudiation, that is, the receiver cannot convince a third party that the message was from the claimed sender.

If we assume that the time between the sender and the receiver are loosely synchronised and the pseudorandom function $\mathcal{H}(\cdot)$ and the MAC are secure, then an analogous proof as in Perrig et al. (2000) could be used to show that the above authenticated broadcast channel is secure. Note that we say that a *pseudorandom function* $\mathcal{H}(\cdot)$ is secure if the function family $f_k(x) = \mathcal{H}(k, x)$ is a pseudorandom function family in the sense of Goldreich et al. (1987) when k is chosen randomly. That is, a function family $\{f_k(\cdot)\}$ is pseudorandom if the adversary with polynomially bounded resources cannot distinguish between a random chosen function from $\{f_k(\cdot)\}$ and a totally random function with non-negligible probability. We say that a message authentication scheme MAC is secure if a polynomially bounded adversary will not succeed with non-negligible probability in the following game. A random l-bits key k is chosen by the user. The adversary chooses messages $m_1, \ldots, m_t$ and the user generates the MAC codes on these messages using the key k. The adversary succeeds if she could then generate a MAC code on a different message $m' \neq m_1, \ldots, m_t$.

Though the time synchronisation between the sender and the receiver plays an important role in the security of the protocol, they do not need to have 100% accurate clocks. If their clocks are sufficiently accurate, then time synchronisation protocol could be designed to synchronise their clocks to meet the security requirements. The time synchronisation protocols could be based on the point-to-point secure channels discussed in the previous section.

4.5 Authenticated emergency channels

In our basic authenticated broadcast protocol, the receiver cannot verify the authenticity of the message immediately since it needs to wait for the disclosure of the key after a time period of $d\delta$. This is not acceptable for some broadcast messages such as an emergency shutdown. In order to overcome this challenge, the sender may reveal the key used for emergency messages immediately or shortly after the message broadcast. This will open the door for an adversary to modify the emergency messages. For example, if the message passes through a node D before it reaches a node C, D can discard the message and create a different emergency message and forward it to C. In another case, an attacker may jam the target C during the emergency broadcast period and sends C a different emergency message (authenticated using the revealed key for the emergency message) later. However, these attacks are generally not practical since if the bad guy jams the channel in a wireless environment, then he jams himself and he cannot receive the authenticated broadcast message either.

4.6 Authenticated emergency channels with finitely many messages

In this section we design authenticated emergency channels which can only broad finitely many emergency messages. Assume that emergency messages are $e_1, \ldots, e_u$. Without loss of generality, we may assume that $e_i = i$ for $i \leq u$. Before the sender could authentically broadcast these messages, it needs to carry out a commitment protocol.

Let υ be a fixed number. During the message commitment procedure, the sender chooses υ random numbers $N_1^i, \ldots N_\upsilon^i$ for each i $\leq u$. It then computes $r_{i,j} = \mathcal{H}(e_i \| N_j^i)$ for all $i \leq u$ and $j \leq \upsilon$. Using the authenticated broadcast channel, the sender broadcasts the commitments $\{r_{i,j} : i \leq u \text{ and } j \leq \upsilon\}$ to all receivers. Receivers store these commitments in their memory space.

Each time when the sender wants to broadcast the message e_i to receivers emergently, it chooses a random unused $j \leq \upsilon$, and broadcasts (e_i, j, N_j^i) to all receivers. The receiver verifies that $r_{i,j} = \mathcal{H}(e_i \| N_j^i)$. If the verification is successful, it knows that the message e_i comes from the sender and delivers it to the target. At the same time, the receiver deletes the commitment $r_{i,j}$ from its memory space.

Note that after each message commitment procedure, the sender could broadcast each message at most υ times. Thus the sender may decide to initiate the message commitment protocol when any one of these messages has been broadcast sufficiently many times (e.g., $\upsilon - 1$ times). Each time when the message commitment protocol is initiated, both the sender and the receiver should delete all previous commitments from their memory space.

The security of the emergency channel could be proved formally under the assumption that the pseudorandom function $\mathcal{H}(\cdot)$ is a secure one-way function. That is, for any given y with appropriate length, one cannot find an x such that $\mathcal{H}(x) = y$ with non-negligible probability.

Theorem 4.1 Assume that the authenticated broadcast channel is secure and the pseudorandom function $\mathcal{H}(\cdot)$ is a secure one-way function. Then the authenticity of messages that receivers accept from the emergency channel is guaranteed.

Sketch of proof. Assume for a contradiction that the authenticity of the emergency protocol is broken. That is, there is an adversary $\mathcal{A}$ who controls communication links and manages to deliver a message m to the receiver such that the sender has not sent the message but the receiver accepts the message. We show in the following that then $\mathcal{H}(\cdot)$ is not a secure one-way function. Specifically, let t be the total number of messages that the sender can broadcast in the emergency channel with one commitment $\{r_{i,j}\}$, and $y_1, \ldots, y_t$ be t randomly chosen strings with appropriate lengths (i.e., they are potential outputs of $\mathcal{H}$). We will construct an algorithm $\mathcal{P}$ that uses $\mathcal{A}$ to compute a pre-image $x = \mathcal{H} - 1(y_i)$ of some string y_i with non-negligible probability.

Since the broadcast channel is secure, we can always assume that the commitment $\{r_{i,j}\}$ that the receivers accept are authentic. The algorithm $\mathcal{P}$ works by running $\mathcal{A}$ as follows. Essentially, $\mathcal{P}$ simulates an authenticated broadcast channel for $\mathcal{A}$ with a sender A and a receiver B.

1 $\mathcal{P}$ chooses a random number $l \le t$.

2 $\mathcal{P}$ computes a commitment $\{r_{i,j}\}$ as specified in the emergency broadcast protocol. $\mathcal{P}$ picks $t - l + 1$ random values from the commitment $\{r_{i,j}\}$ and replace them with $y_l, y_{l+1}, \ldots, y_t$.

3 $\mathcal{P}$ runs the sender's algorithm to authentically broadcast the modified commitment to B.

4 For the first $l - 1$ emergency messages, $\mathcal{P}$ runs the sender's algorithm of the emergency broadcast protocol with no modification to broadcast the pre-images of the $l - 1$ unmodified commitments.

5 $\mathcal{P}$ then waits for $\mathcal{A}$ to deliver a fake message x' that B accepts as an authentic emergency broadcast. $\mathcal{P}$ outputs x' as one of the pre-images of $y_l, \ldots, y_t$.

We briefly argue that $\mathcal{P}$ outputs the pre-image of one of the strings from $y_1, \ldots, y_t$ with non-negligible probability. Since $\mathcal{A}$ succeeds with non-negligible probability in convincing the receiver to accept a fake message, it must deliver this message as the l-th message for some $l \le t$ in the authenticated emergency channel. Thus for this l, the algorithm $\mathcal{P}$ outputs a pre-image for one of the given strings with non-negligible probability. QED.

Theorem 4.1 shows that messages received in the emergency channel are authentic. However, it does not show whether these messages are fresh. Indeed, when the sender broadcasts an emergency message at the time t, the adversary may launch a denial of service attack against the receiver or just does not deliver the message to the receiver. Thus the receiver will not be able to delete the commitment of this message from its memory space. Later at time t', the adversary delivers this message to the receiver and the receiver accepts it. In our emergency channel, there is no way to avoid this kind of delayed message attacks. Thus when message freshness is important, one may use the revised authenticated emergency broadcast channel that we will discuss in the next section.

4.7 Revised authenticated emergency channel

There are basically two ways to guarantee the freshness of a received message. The first one is to use public key cryptography together with time-stamps. The second solution is to let the receiver send a nonce to the sender first and the sender authenticates the message together with the nonce. As we have mentioned earlier, public key cryptography is too expensive to be deployed in SCADA systems. For the second solution, the delays introduced in once submission process are generally not acceptable in an emergent situation. In this section, we introduce a revised emergency broadcast protocol, which provides weak freshness of received messages. Here weak freshness means that the received message is guaranteed to be in certain time limit T. In another word, at time t, the adversary cannot convince a receiver to accept a message that is posted before the time $t - T$.

Let the u emergency messages be $e_1, \ldots, e_u$. Similar to the previous protocol, the sender needs to carry out a commitment protocol before the authenticated emergency broadcast. In the revised protocol, the sender chooses v random numbers $N_1^i, \ldots N_\upsilon^i$ and υ expiration time points $T_1^i < T_2^i < \ldots T_\upsilon^i$ for each $i \leq u$. It then computes $r_{i,j} = \mathcal{H}(e_i \| N_j^i \| T_j^i)$ for $i \leq u$ and $j \leq \upsilon$. Using the authenticated broadcast channel, the sender broadcasts the commitments $\{r_{i,j} : i \leq u \text{ and } j \leq \upsilon\}$ to all receivers. Receivers store these commitments in their memory space. The functionality of expiration time points in the revised protocol is to guarantee that the commitment $r_{i,j}$ for the message e_i expires at the time T_j^i. In another word, when the receiver receives (e_i, N_j^i, T_j^i), it will accepts the message only if the current clock time of the receiver is earlier than T_j^i.

If the sender wants to send the message e_i to receivers at time t, it chooses a random unused $j \leq \upsilon$ such that $t < T_j^i$, the estimated transmission time from the sender to receiver is less than $T_j^i - t$, and T_j^i is the earliest time in the commitments that satisfies these conditions. Then the sender broadcasts (e_i, j, N_j^i, T_j^i) to all receivers. The receiver verifies that $r_{i,j} = \mathcal{H}(e_i \| N_j^i \| T_j^i)$ and the current clock time of the receiver is earlier than T_j^i. If the verification is successful, it knows that the message e_i comes from the sender and delivers it to the target. At the same time it deletes the commitment $r_{i,j}$ from its memory space. Otherwise, the receiver discards the message.

The implementation of the revised emergency broadcast protocol has the flexibility to choose the gaps between expiration time points T_j^is for each $i \leq u$. The smaller the gap, the better the freshness property. However, smaller gaps between T_j^is add additional overhead on the communication links. It is also possible, for different messages e_i, one chooses different values υ. For example, for more frequently broadcast message, the value of υ should be larger. It is also important to guarantee that the commitment is always sufficient and when only a few commitments are unused, the sender should initiate a procedure for a new commitment.

The security of the revised emergency broadcast protocol can be proved similarly as in Theorem 4.1. It is still possible for an adversary to delay an emergency message (e_i, j, N_j^i, T_j^i) broadcast by the sender during the time period $[T_{j-1}^i, T_j^i]$ until T_j^i. However, she cannot delay the message to some time points after T_j^i. In another word, weak freshness of received messages are guaranteed in the revised authenticated emergency channel.

5 Conclusions

In this paper, we systematically discussed the security issues for SCADA systems and the challenges to design such a sSCADA system. In particular, we present an attack on the protocols in the first version of AGA standard draft (Wright et al., 2004). This attack shows that the security mechanisms in the first draft of the AGA standard protocol could be easily defeated. We then proposed a suite of security protocols optimised for SCADA/DCS systems which include: point-to-point secure channels, authenticated broadcast channels, authenticated emergency channels, and revised authenticated emergency channels. These protocols are designed to address the specific challenges that SCADA systems have.

Recently, there has been a wide interest for the secure design and implementation of smart grid systems (DOE, 2009). SCADA system is one of the most important legacy systems of the smart grid systems. Together with other efforts such as Idaho National Laboratory (2008), NIST (2008), IEEE P1711, IEEE P1815, IEC TC57 and IEC 60870-5, our work in this paper presents an initial step for securing the SCADA section of the smart grid systems against cyber attacks.

Acknowledgements

The author would like to thank the anonymous referees for the excellent comments on the improvement of this paper.

References

Abrams, M. and Weiss, J. (2007) 'Bellingham, Washington, control system cyber security case study', available at http://csrc.nist.gov/groups/SMA/fisma/ics/documents/Bellingham Case Study report%2020Sep071.pdf (accessed on 22 February 2010).

Abrams, M. and Weiss, J. (2007) 'Malicious control system cyber security attack case study – Maroochy water services', Australia, available at http://csrc.nist.gov/groups/SMA/fisma/ics/documents/Maroochy-Water-Services-Case-Study briefing.pdf (accessed on 22 February 2010).

AGA Report No. 12 (2004) 'Cryptographic protection of SCADA communications: general recommendations', Draft 2, 5 February 2004, The draft 2 is no longer available online, The draft 3 is available for purchase at http: //www.aga.org/ (accessed on 22 February 2010).

Anderson, R., Bergadano, F., Crispo, B., Lee, J., Manifavas, C. and Needham, R. (1998) 'A new family of authentication protocols', *Operating Systems Review*, October, Vol. 32, No. 4, pp.9–20.

Bellare, M., Boldyreva, A., Knudsen, L. and Namprempre, C. (2001) 'On-line ciphers and the Hash-CBC constructions', *Advances in Cryptology – Crypto 2001*, Lecture Notes in Computer Science 2139, Springer-Verlag.

Bellare, M., Canetti, R. and Krawczyk, H. (1996) 'Message authentication using hash functions – the HMAC construction', *RSA Laboratories CryptoBytes*, Spring, Vol. 2, No. 1.

Bellare, M., Kilian, J. and Rogaway, P. (2000) 'The security of the cipher block chaining message authentication code', *Journal of Computer and System Sciences*, Vol. 6, No. 3, pp.362–399.

Bergadano, F., Cavagnino, D. and Crispo, B. (2000a) 'Chained stream authentication', *Selected Areas in Cryptography*, Waterloo, Canada.

Bergadano, F., Cavagnino, D. and Crispo, B. (2000b) 'Individual single source authentication on the mbone', *ICME 2000*, August, pp.541–544.

Blumenfeld, L. (2003) 'Dissertation could be security threat', *Washington Post*, available at http://www.washingtonpost.com/ac2/ wp-dyn/A23689-2003Jul7 (accessed on 22 February 2010).

Briscoe, B. (2000) 'FLAMeS: fast, loss-tolerant authentication of multicast streams', Technical report, available at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.9263&rep=rep1&type=pdf (accessed on 22 February 2010).

Cegrell, T. (1986) *Power System Control Technology*, Prentice-Hall International (UK) Ltd.

Cheung, S. (1997) 'An efficient message authentication scheme for link state routing', *13th Annual Computer Security Applications Conference*.

DOE (2009) 'Study of security attributes of smart grid systems – current cyber security issues', April, available at http://www.inl.gov/scada/publications/d/securing_the_smart_grid_current_issues.pdf (accessed on 22 February 2010).

GAO-04-628T (2004) 'Critical infrastructure protection: challenges and efforts to secure control systems', *Testimony Before the Subcommittee on Technology Information Policy, Intergovernmental Relations and the Census, House Committee on Government Reform*, 30 March, available at http://www.gao.gov/new.items/d04628t.pdf (accessed on 22 February 2010).

Goldreich, O., Goldwasser, S. and Micali, S. (1987) 'How to construct random functions', *J. of the ACM*, Vol. 33, No. 4, pp.792–807.

Goldwasser, S. and Michali, S. (1984) 'Probabilistic encryption', *Journal of Computer and System Sciences*, Vol. 28, pp.270–299.

Haller, N. (1995) 'The S/KEY one-time password system', *IETF RFC 1760*, February.

Idaho National Laboratory (2008) 'SCADA test bed program', available at http://www.inl.gov/scada/ (accessed on 22 February 2010).

IEC 608705, 'User group maillist information', available at http://www.trianglemicroworks.com/iec60870-5/index.htm (accessed on 22 February 2010).

IEC TC57, available at http://www.ieee.org/portal/cms_docs_pes/pes/subpages/publications-folder/TC_57_Column.pdf (accessed on 22 February 2010).

IEEE P1711, 'Trial use standard for a cryptographic protocol for cyber security of substation serial links', available at http://grouper.ieee.org/groups/sub/wgc6/documents/drafts/IEEE%20P1711%20Draft2006-12-10.doc (accessed on 22 February 2010).

IEEE P1815, 'Standard for electric power systems communications – distributed network Protocol (DNP3).

Krawczyk, H., Bellare, M. and Canetti, R. (1997) 'HMAC: keyed-hashing for message authentication', *Internet RFC 2104*, February.

Lamport, L. (1981) 'Password authentication with insecure communication', *Commun. ACM*, Vol. 24, No. 11.

NIST (1981) 'DES model of operation', *FIPS Publication 81 (FIPS PUB 81)*.

NIST (2008) 'Industrial control system security (ICS)', available at http://csrc.nist.gov/groups/SMA/fisma/ics/index.html (accessed on 22 February 2010).

Perrig, A., Canetti, R., Song, D. and Tygar, J. (2001) 'Efficient and secure source authentication for multicast', *Network and Distributed System Security Symposium, NDSS'01*.

Perrig, A., Canetti, R., Tygar, J. and Song, D. (2000) 'Efficient authentication and signing of multicast streams over lossy channels', *IEEE Symposium on Security and Privacy*.

Perrig, A., Szewczyk, R., Tygar, J., Wen, V. and Culler, D. (2002) 'SPINS: security protocols for sensor networks', *Wireless Networks*, Vol. 8, pp.521–534.

Rappaport, J. (2007) 'What you don't know might hurt you: Alum's work balances national security and information sharing', available at http://gazette.gmu.edu/articles/11144 (accessed on 22 February 2010).

SCADAsafe, available at http://scadasafe.sourceforge.net.

USA Today (2007) 'AURORA case: US video shows hacker hit on power grid', available at http://www.usatoday.com/tech/news/computersecurity/2007-09-27-hacker-video_N.htm (accessed on 31 May 2010).

Wright, A.K., Kinast, J.A. and McCarty, J. (2004) 'Low-latency cryptographic protection for SCADA communications', *Proc. 2nd Int. Conf. on Applied Cryptography and Network Security, ACNS 2004*, LNCS 3809, Springer, pp.263–277.